Stable evaluation of 3D Zernike moments for surface meshes

Jérôme Houdayer, Patrice Koehl

Stable evaluation of 3D Zernike moments for surface meshes

Jérôme Houdayer
Université Paris-Saclay, CNRS, CEA,
Institut de Physique Théorique, F-91191 Gif-sur-Yvette, France
e-mail: jerome.houdayer@ipht.fr

Patrice Koehl
Department of Computer Science,
University of California, Davis, CA 95616.
e-mail: koehl@cs.ucdavis.edu

September 1, 2022

Abstract

The 3D Zernike polynomials form an orthonormal basis of the unit ball. The associated 3D Zernike moments have been successfully applied for 3D shape recognition; they are popular in structural biology for comparing protein structures and properties. Many algorithms have been proposed for computing those moments, starting from a voxel-based representation or from a surface based geometric mesh of the shape. As the order of the 3D Zernike moments increases, however, those algorithms suffer from decrease in computational efficiency and more importantly from numerical accuracy. In this paper, new algorithms are proposed to compute the 3D Zernike moments of a homogeneous shape defined by an unstructured triangulation of its surface that remove those numerical inaccuracies. These algorithms rely on the analytical integration of the moments on tetrahedra defined by the surface triangles and a central point and on a set of novel recurrent relationships between the corresponding integrals. The mathematical basis and implementation details of the algorithms are presented and their numerical stability is evaluated. We show that moments up to order 300 can be computed with a finite precision of 10^{-10}. The corresponding free software is available at `https://github.com/jerhoud/zernike3d`.

1 Introduction

Finding efficient algorithms to describe, measure and compare shapes is a central problem in image processing. This problem arises in numerous disciplines that generate extensive quantitative and visual information. Among these, biology occupies a central place [1]. In cellular biology for example, the measurements of cell morphology and dynamics by imaging techniques such as light or fluorescent microscopy yield large amounts of 2D and 3D images that need to be segmented and quantified [2–5]. Measuring shapes, computing the contribution of a shape to a potential, and more generally quantifying the effect of a vector field on a shape are also common problems for geodesists and physicists.

In a chapter titled "The Comparison of Related Forms", Thompson explored how differences in the forms of related animals can be described by means of simple mathematical transformations [6]. This inspired the development of several shape comparison techniques, whose aim is to define a map between two shapes that can be used to measure their similarity. An alternate and popular method is to derive features (also called shape descriptors or signatures) for each shape separately that can then be compared using standard distance functions, and those that

directly attempt to map one surface onto the other, thereby providing both local and non-local elements for comparison. In this paper, we are concerned with this approach.

One particular powerful technique for generating shape signatures is based on moment-based representations of a shape. Those representation form a class of shape recognition techniques that have been used widely for pattern recognition [7–9]. These moments not only provide measures of the shapes, such as volume and surface areas, they also allow for the encoding of a shape with descriptors that are amenable to fast analysis. The most common of these moments are geometric:

$$G_{ijk} = \int_V f(x,y,z) x^i y^j z^k \mathrm{d}V \tag{1}$$

where the integration is performed over the volume V of the shape S considered, $N = i + j + k$ is the order of the moment, and $f(x, y, z)$ is a vector field over S that may represent a potential, a grey scale for image processing, or an indicator function such that $f(x, y, z) = 1$ inside the shape and 0 otherwise. These geometric moments and their invariant extensions have been used extensively in pattern recognition (for review, see for example [10]). They are usually easy to compute, though at a high computational costs.

Spherical harmonics [11,12] and their rotational invariants [13] form another class of moment-based descriptors for analyzing star-shaped objects that are topologically equivalent to a sphere. They are becoming increasingly popular in cellular bio-imaging for example, as it is usually safe to assume that cells observed through a microscope are topologically closed and equivalent to a sphere; this prior makes it possible to parametrize their shape mathematically, thereby simplifying the definition of their surface and provide a shape description (see for example [3] for representing cell organelles and more recently [14] for studies of cell dynamics).

In many cases however, the hypothesis that the object is star-shaped does not hold. The Zernike moments, first introduced in two dimensions by Dutch Nobel price Frederic Zernike in 1934 [15], circumvent this problem through the introduction of a radial term. They have proved to be superior in 2D image retrieval (see for example [16]). After they were generalized to 3D by Canterakis [17], they have been applied in many domains, such as tools for shape retrieval in computer graphics [18, 19] or terrain matching and building reconstruction [20, 21],. As for geometric moments, most current applications are related to biology, including protein shape comparison [22], protein docking [23] and the analyses of protein interfaces [24–26]. In this paper, we are concerned with the computation of these 3D Zernike moments.

Many algorithms have been proposed for computing the 3D Zernike moments of a shape. Most of these methods, especially those used for image analysis, rely on a representation of the shape as a volumetric grid. These algorithms usually proceed in two steps, namely computing the geometric moments of the shape first, and then expressing the Zernike moments as linear combinations of those moments (see Refs [18,27], and background section below). They do suffer from three major drawbacks. First, the computation of each moment has a cubic computational complexity (N_V^3, where N_V is the number of voxels in each dimension of the grid). Second, the volume integral in equation 1 is approximated by a discrete sum over the voxels, where each voxel contributes as a point-like object, usually located at its center. It is easy to see that this discretization error increases as the order of the moment increases. Finally, the relationships between geometric moments and Zernike moments lead to numerical instabilities, limiting the order N of the moments that can be computed accurately, usually with $N < 50$ (see [18, 28]).

While it is often the case that the volumetric grid is the only representation available for a given shape, alternate methods that are exact and efficient exist for computing homogeneous 3D moments over a shape whose boundary can be represented with a polygonal mesh (where homogeneity refers to the fact that the function $f(x, y, z)$ in equation 1 can be considered constant). An exact formula for computing geometric moments on 3D polyhedra was originally proposed by Lien and Kajiya [29]; it has been reformulated and extended to general dimensional polyhedra several times since then (for a complete discussion of the state of the art in this field, see [30]). Pozo et al. used those ideas and derived efficient recursive algorithms for computing geometric moments for shapes defined by a triangulation of their surfaces [30]; those algorithms were further refined and optimized to be optimal with respect to the order of the

moments [31]. Pozo et al. then used those geometric moments to derive the 3D Zernike moments of such shapes. As mentioned above, however, the numerical instabilities associated with the relationships between geometric and Zernike moments limit the order with which those can be derived.

Recently Deng and Gwo proposed a new, stable algorithm to accurately compute Zernike moments for shapes represented as 3D grid [32]. Their algorithm proceeds by computing these moments directly, without using the geometric moments, using recurrence relations that provide stability and efficiency. Our work presented in this paper is a counterpart to the work of Deng and Gwo. We propose a new exact algorithm for the computation of Zernike moments for shapes represented by surface meshes. Similar to Deng and Gwo, we do not use the conversion from geometric moments to Zernike moments to avoid numerical instabilities.

The paper is organized as follows. In the next section, we give some background on moments of 3D shapes, especially geometric moments and Zernike moments. We show how the former can be used to compute the latter, but explain why this may lead to numerical instabilities, especially for high order moments. The following section introduce our new, stable algorithms for computing Zernike moments of 3D shapes represented by surface-based triangular meshes. The results section introduce some experiments to validate both algorithms on synthetic data.

2 Moments from 3D shapes

In this section, we briefly introduce the concept of moments of a shape, with three examples, the geometric moments (GM), the spherical harmonics moments (SPHM), and the Zernike moments (ZM). We show that ZM are extensions of the SPHM, and that they can be computed from the GM, but with potential problems that we describe.

We start with notations. Any point $\boldsymbol{p}$ within a domain $D \subset \mathbb{R}^3$ can be parametrized either in terms of its vector of Cartesian coordinates $\boldsymbol{x} = (x, y, z)$ with respect to an origin, or in terms of its spherical coordinates (r, θ, ϕ) where r is the distance from $\boldsymbol{p}$ to the sphere center, and θ and ϕ are the inclination and azimuthal angles, respectively. A shape S in the domain D is represented through its density $f(\boldsymbol{x})$ at all points in the domain. We assume that f is square integrable over the domain D, i.e. $f \in L^2$. In the special case that the shape is homogeneous, it is represented with an indicator function $f_S(\boldsymbol{x})$ with value 1 if $\boldsymbol{x}$ is inside the shape, and 0 otherwise.

2.1 Moments of a shape

The moments μ_i of a shape are the projections of the function f over a set of basis functions $\boldsymbol{\Psi} = \{\psi_i\}$:

$$\mu_i = \langle f, \psi_i \rangle = \int_D f(\boldsymbol{x}) \psi_i(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

The properties of a particular moment based representation are therefore determined by the set of functions $\boldsymbol{\Psi}$. There are two properties of this set that are desirable, namely:

i) *Orthonormality.* The set of functions $\boldsymbol{\Psi}$ is orthonormal if

$$\langle \psi_i, \psi_j \rangle = \delta_{ij}$$

for all ψ_i and ψ_j in $\boldsymbol{\Psi}$ (δ is the Kronecker symbol, i.e. $\delta_{i,j} = 1$ if $i = j$, and zero otherwise).

ii) *Completeness.* The set of functions $\boldsymbol{\Psi}$ is complete if for all functions $f \in L^2$,

$$\lim_{n \to +\infty} ||f - \sum_{i=0}^{n} \langle f, \psi_i \rangle \psi_i|| = 0$$

.

The orthonormality property is important as it guarantees the mutual independence of the computed moments. The completeness property implies that we are able to reconstruct approximations of the original shape from its moments.

2.2 Basis function: monomials

A very popular set of functions $\mathbf{\Psi}$ are the monomials $x^i y^j z^k$, where i, j, k are positive integers. The corresponding moments are referred to as geometric moments, G_{ijk}, and defined by

$$G_{ijk} = \int_D f(\boldsymbol{x}) x^i y^j z^k \mathrm{d}\boldsymbol{x}, \tag{2}$$

The geometric moments are easy to compute, both for grid-based and surface-based representations of shapes. The geometric monomials, however, are neither orthonormal, nor complete.

2.3 Basis function: Laplace spherical harmonics

If the domain is the sphere $\mathbb{S}^2 \subset \mathbb{R}^3$, a point $\boldsymbol{p}$ is characterized by its inclination angle θ and azimuthal angle ϕ only. A function f on $\mathbb{S}^2$ can then be represented through its spherical harmonics. They form a Fourier basis on a sphere much like the familiar sines and cosines do on a line. The spherical harmonic Y_l^m is defined by

$$Y_l^m(\theta, \phi) = N_l^m P_l^m(\cos\theta) e^{im\phi} \tag{3}$$

where N_l^m is a normalization factor:

$$N_l^m = \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}}$$

and P_l^m are the associated Legendre polynomial. Y_l^m is defined for l positive integer, and m integer such that $-l \leq m \leq l$. It is enough, however, to compute the spherical harmonics for $m \geq 0$, as we have the relationship,

$$\overline{Y_l^m(\theta, \phi)} = (-1)^m Y_l^{-m}(\theta, \phi)$$

where $\overline{Y_l^m}$ stands for the complex conjugate of Y_l^m. We note the definition of the harmonic polynomial,

$$e_l^m(\boldsymbol{x}) = r^l Y_l^m(\theta, \phi) \tag{4}$$

where $\boldsymbol{x}$ is the point with spherical coordinates (r, θ, ψ).

The spherical harmonics moments f_l^m are defined as:

$$f_l^m = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} f(\theta, \phi) \overline{Y_l^m(\theta, \phi)} \sin\theta \mathrm{d}\phi \mathrm{d}\theta. \tag{5}$$

Note that these moments are complex, as the spherical harmonics are complex.

The spherical harmonics are orthonormal:

$$\int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} Y_l^m(\theta, \phi) \overline{Y_{l'}^{m'}(\theta, \phi)} \sin\theta \mathrm{d}\phi \mathrm{d}\theta = \delta_{mm'} \delta_{ll'} \tag{6}$$

where δ_{mn} is the Kronecker delta (i.e. $\delta_{mn} = 1$ if $m = n$ and 0 otherwise).

Just like Fourier series are complete, the spherical harmonics are complete on the sphere. It is important to notice, however, that they cannot be computed on a volumetric shape without some modifications, such as the use of solid harmonics, or with the introduction of Zernike polynomials, which are presented in the following subsection.

2.4 Basis function: the Zernike polynomials

The spherical harmonics are defined on the sphere. Zernike [15] in 2D and later Canterakis [17] in 3D have shown that they can be expanded to account for the whole ball by using the Zernike polynomials. Let $\boldsymbol{x}$ be a point inside the unit ball $\mathbb{B}$ with spherical coordinates (r, θ, ϕ). The value of Zernike polynomial Z_{nl}^m at $\boldsymbol{x}$ is given by:

$$Z_{nl}^m(\boldsymbol{x}) = \sqrt{2n+3} R_{nl}(r) Y_l^m(\theta, \phi) \tag{7}$$

where Y_l^m are the spherical harmonics define above, and R_{nl} are polynomials in the radial coordinate r:

$$R_{nl}(r) = \sum_{\nu=0}^{k} Q_{nl\nu} r^{2\nu+l} \tag{8}$$

where

$$Q_{nl\nu} = (-1)^{k+\nu} \frac{\binom{2k}{k}\binom{k}{\nu}\binom{2(k+l+\nu)+1}{2k}}{4^k \binom{k+l+\nu}{k}} \tag{9}$$

The positive integer n is the order the of Zernike polynomial, l is an integer that is restricted so that $0 \leq l \leq n$ and $(n-l)$ be an even number, $k = (n-l)/2$, and $-l \leq m \leq l$. The coefficients in R_{nl} were chosen to guarantee that the Zernike polynomials are orthonormal, a property expressed in the following equation:

$$\int_{\mathbb{B}} Z_{nl}^m(\boldsymbol{x}) \overline{Z_{n'l'}^{m'}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} = \delta_{nn'} \delta_{ll'} \delta_{mm'} \tag{10}$$

The Zernike moments of a shape S whose density inside the unit ball is defined by the square integrable function f are then defined as

$$\Omega_{nl}^m = \int_{\mathbb{B}} f(\boldsymbol{x}) \overline{Z_{nl}^m(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{11}$$

Note that m can be positive or negative, as it belongs to $[-l, l]$. It is enough, however, to compute the moments for m positive as we have (see for example [18]):

$$\Omega_{nl}^{-m} = (-1)^m \overline{\Omega_{nl}^m} \tag{12}$$

Reconstruction. The Zernike polynomial form a complete basis of L^2. It is therefore possible to approximate the original function f by a finite number of 3D Zernike moments:

$$\hat{f}(\boldsymbol{x}) = \sum_{n}^{N} \sum_{l} \sum_{m=-l}^{l} \Omega_{nl}^m Z_{nl}^m(\boldsymbol{x}) \tag{13}$$

where l is an integer that is restricted so that $0 \leq l \leq n$ and $(n-l)$ be an even number. The approximation is exact when $N \to +\infty$.

2.5 Computing the Zernike polynomials from the Geometric moments

Although the Zernike polynomials are usually defined with respect to spherical coordinates, they are actually polynomial functions on Cartesian coordinates (x, y, z). To show this, let us first rewrite the Zernike polynomials,

$$\begin{aligned} Z_{nl}^m(\boldsymbol{x}) &= \sqrt{2n+3} R_{nl}(r) Y_l^m(\theta, \phi) && (14) \\ &= \sqrt{2n+3} \sum_{\nu=0}^{k} Q_{nl\nu} (x^2+y^2+z^2)^\nu r^l Y_l^m(\theta, \phi) && (15) \\ &= \sqrt{2n+3} \sum_{\nu=0}^{k} Q_{nl\nu} (x^2+y^2+z^2)^\nu e_l^m(\boldsymbol{x}) && (16) \end{aligned}$$

where $k = (n-l)/2$ and e_l^m is the harmonic polynomial (see equation 4) and $\boldsymbol{x} = (x, y, z) = (r, \theta, \phi)$. The harmonic polynomial can be expressed in Cartesian coordinates [18]:

$$e_l^m(\boldsymbol{x}) = c_l^m r^l (\frac{ix-y}{2})^m \sum_{\mu=0}^{\lfloor \frac{l-m}{2} \rfloor} (-1)^\mu \binom{l}{\mu} \binom{l+\mu}{m+\mu} (r^2 - z^2)^\mu z^{l-m-2\mu}$$

where c_l^m is a normalization factor,

$$c_l^m = \frac{\sqrt{(2l+1)(l+m)!(l-m)!}}{l!}$$

The Zernike polynomial Z_{nl}^m can then be expressed as:

$$Z_{nl}^m(\boldsymbol{x}) = \sqrt{2n+3} \frac{c_l^m}{2^m} \sum_{\nu=0}^{k} Q_{nl\nu} \sum_{\alpha=0}^{\nu} \binom{\nu}{\alpha} \sum_{\beta=0}^{\nu-\alpha} \binom{\nu-\alpha}{\beta} \sum_{u=0}^{m} (-1)^{m-u} \binom{m}{u} i^u \times$$
$$\sum_{\mu=0}^{\lfloor \frac{l-m}{2} \rfloor} (-1)^\mu \binom{l}{\mu} \binom{l+\mu}{m+\mu} \sum_{v=0}^{\mu} \binom{\mu}{v} x^{2(v+\alpha)+u} y^{2(\mu-v+\beta)+m-u} z^{2(\nu-\alpha-\beta-\mu)+l-m} \quad (17)$$

We set $r = 2(v+\alpha+u)$, $s = 2(\mu-v+\beta)+m-u$, $t = 2(\nu-\alpha-\beta-\mu)+l-m$, and

$$\chi_{nlm}^{rst} = \sqrt{2n+3} \frac{c_l^m}{2^m} \sum_{\nu=0}^{k} Q_{nl\nu} \sum_{\alpha=0}^{\nu} \binom{\nu}{\alpha} \sum_{\beta=0}^{\nu-\alpha} \binom{\nu-\alpha}{\beta} \sum_{u=0}^{m} (-1)^{m-u} \binom{m}{u} i^u \times$$
$$\sum_{\mu=0}^{\lfloor \frac{l-m}{2} \rfloor} (-1)^\mu \binom{l}{\mu} \binom{l+\mu}{m+\mu} \sum_{v=0}^{\mu} \binom{\mu}{v} \delta_{r,2(v+\alpha+u)} \delta_{s,2(\mu-v+\beta)+m-u} \delta_{t,2(\nu-\alpha-\beta-\mu)+l-m}. \quad (18)$$

Note that $r+s+t = 2\nu+l \leq 2k+l = n$. Replacing in the expression for the Zernike polynomial, we have

$$Z_{nl}^m(\boldsymbol{x}) = \sum_{r+s+t \leq n} \chi_{nlm}^{rst} x^r y^s z^t \quad (19)$$

Finally, after replacing the Cartesian expression for Z_{nl}^m (equation 17) into equation (11), we get:

$$\Omega_{nl}^m = \sum_{r+s+t \leq n} \chi_{nlm}^{rst} G_{rst}$$

where G refers to the geometric moment (see equation 2). This allow for a simple algorithm for computing the Zernike moments of a shape from its geometric moments. Similar algorithms have been proposed [18, 30] for the same task. Such an algorithm is theoretically very efficient, once the geometric moments have been computed, as it is independent of the size of the mesh representing the shape or the number of grid point in a voxel representation of the shape. There are, however, numerical issues with this formula that we discuss below.

2.6 Numerical instabilities associated with the Zernike polynomials

Let us first look at the radial polynomials. R_{nl} is a polynomial of degree n. For large values of n, special care is needed for computing them, and direct application of Equation 8 is bound to numerical instabilities, as described in figure 1.

The values of $R_{80,0}(r)$ as a function of r, based on a stable evaluation of the polynomial function vary in the interval $[-1.57, 7.20]$, where the largest value is reached for $r = 0$. Direct application of Equation 8 shows however, that $R_{80,l}(r)$ varies in the interval $[-4149, 4 \times 10^{13}]$ (results not shown). This is due to the fact that the coefficients $Q_{80,0,\nu}$ are large (up to 10^{29}), as illustrated in panel B of figure 1. Those coefficients alternate from positive to negative due

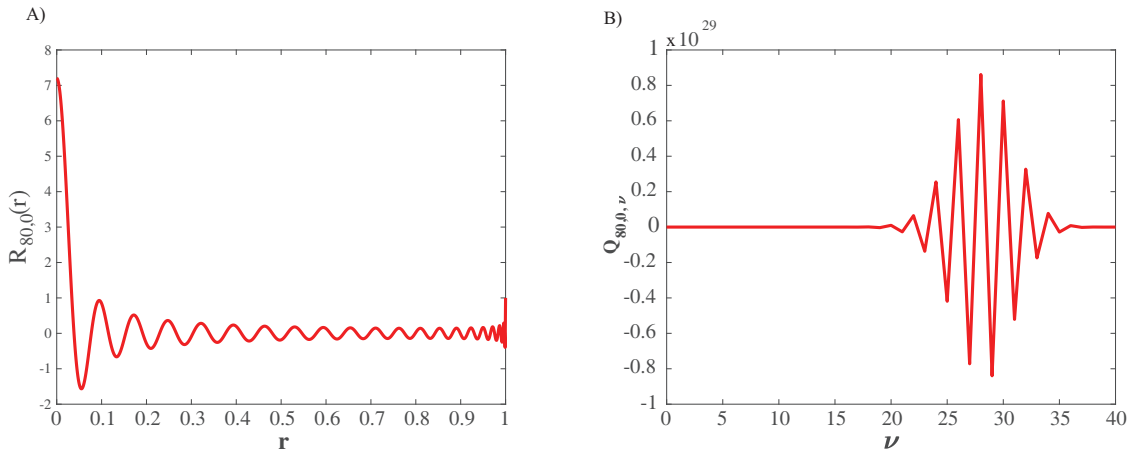

Figure 1: *Instabilities in evaluating the R_{nl} radial polynomials.* In panel A, we show the values of $R_{80,0}(r)$ as a function of r, based on a stable evaluation of the polynomial function (see text for details). In panel B, we show the values of the coefficients $Q_{80,l,\nu}$, the monomial expansion of $R_{80,0}(r)$.

to the presence of the term $(-1)^\nu$, leading to large cancellations and ultimately to a small value for the polynomial. Computing correctly those cancellations requires very high precision usually not available with standard double precision in programming languages. It is possible to use arbitrary precision libraries to solve this issue, but it is in fact not necessary. As was noticed multiple times for the 3D Zernike radial polynomials (see for example [33,34]) the radial polynomial R_{nl} can be expressed as a Jacobi polynomial:

$$R_{nl}(r) = r^l P_{\frac{n-l}{2}}^{(0,l+3/2)}(2r^2 - 1) \tag{20}$$

Equation 20 allows the results available in the literature for the Jacobi polynomials to be translated for the 3D radial functions. In particular, we have the following recurrence relation (it was also derived in [32])

$$R_{nl}(r) = (K_1(n,l)r^2 + K_2(n,l))R_{n-2,l}(r) + K_3(n,l)R_{n-4,l}(r) \tag{21}$$

where the coefficients K_i are defined as:

$$\begin{aligned} k_0 &= (n-l)(n+l+1)(2n-3) \\ k_1 &= (2n-1)(2n+1)(2n-3) \\ k_2 &= \frac{1}{2}(-2n+1)(2l+1)^2 - \frac{k_1}{2} \\ k_3 &= -(n-l-2)(n+l-1)(2n+1) \\ K_1(n,l) &= \frac{k_1}{k_0}, \quad K_2(n,l) = \frac{k_2}{k_0}, \quad K_3(n,l) = \frac{k_3}{k_0} \end{aligned} \tag{22}$$

This recursive formula is only strictly valid for $l \leq n-4$. This can be addressed by noticing that

$$\begin{aligned} R_{nn}(r) &= r^n \\ R_{n,n-2}(r) &= (n+\frac{1}{2})r^n - (n-\frac{1}{2})r^{n-2} \end{aligned} \tag{23}$$

This recurrence allows for a stable computation of all $R_{nl}(r)$, even for large orders.

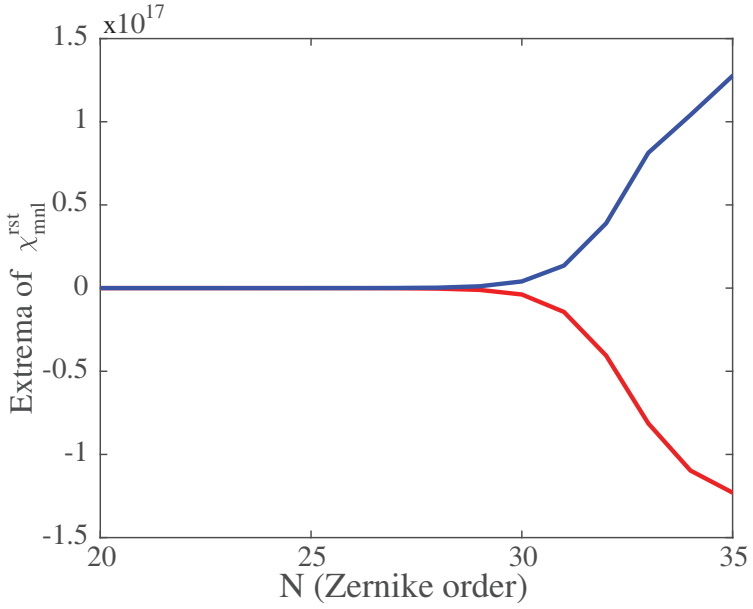

Figure 2: *Instabilities in evaluating the* χ^{rst}_{nlm} *factors.* We show the maxima (in blue) and minima (in red) for the χ^{rst}_{nlm} for a given n, as a function of n.

The geometric moments of a shape can be computed accurately even for large orders, see for example [30, 31]. Those moments can then be used to evaluate the 3D Zernike moments, as described in the section above. Converting the geometric moments to Zernike moments require, however, that the factor χ^{rst}_{nlm} be computed (where nlm refers to the indices for the Zernike moments, while rst refers to the indices for the geometric moments). As defined in 18, the factors χ^{rst}_{nlm} depend on the coefficients $Q_{nl\nu}$ and therefore they are bound to suffer from the same numerical instabilities. We illustrate this in Figure 2.

As expected, the χ^{rst}_{nlm} vary significantly over a large range of values. This was already observed by Berjón and colleagues in their attempts to parallelize the computation of Zernike moments [28] and is the main reason that the computation of Zernike moments is usually limited to order below $N = 40$. One solution to solve this problem would be to derive recurrence relationships for the χ^{rst}_{nlm}. Pozo *et al.* provided such a recurrence. However, their relationships still involve the computations of the factors $Q_{nl\nu}$ (see their equation (13e)) and as such do not solve the numerical instabilities.

Deng and Gwo [32] proposed a different approach in their attempt to compute Zernike moments for a shape described on a grid, which is to bypass the computation of the geometric moments. In the following, we propose a similar approach for computing Zernike moments for a shape described by a surface-based triangular mesh.

3 Algorithms for computing the 3D Zernike moments for a surface triangular mesh

3.1 Zernike moments over a shape defined by a triangular surface mesh

Let us consider a shape S, covering a volume V. The shape is assumed to be represented by a triangle mesh that defines its boundary. Each facet (triangle) T is defined by three vertices, A, B and C that are oriented consistently counter-clockwise when seen from the exterior of the shape. As the Zernike polynomials are complete and orthonormal over the unit ball $\mathbb{B}$, the

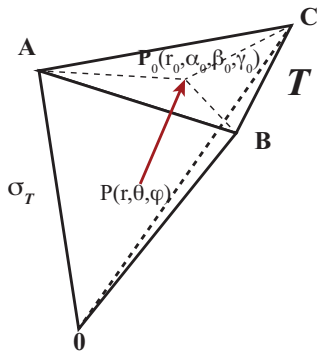

Figure 3: *Parameterization of a tetrahedron.* Let $T = (A, B, C)$ be a triangle in the surface mesh, and $\sigma_T = (0, A, B, C)$ be the associated tetrahedron where 0 is the origin.Any point P within σ_T with spherical coordinates (r, θ, ϕ) is projected the triangle T, setting P_0. P_0 is defined by its distance to O, r_0, and its barycentric coordinates within T, $\alpha_0, \beta_0, \gamma_0$ with $\alpha_0 + \beta_0 + \gamma_0 = 1$.

shape is assumed to fit within this ball. This is obtained by centering the mesh to the origin O, and scaling the mesh such that the longest distance between a vertex of the mesh and O is set to one.

Assuming that this shape is homogeneous (i.e. represented by a constant scalar field, with the constant set to one), its Zernike moments Ω_{nl}^m are given by:

$$\Omega_{nl}^m = \int_V \overline{Z_{nl}^m(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{24}$$

where n is the order of the moment, l is a positive integer smaller or equal to n, with the same parity as n, and m an integer with $-l \leq m \leq l$..

Using the origin O of the coordinate frame as a reference point, each facet T defines a tetrahedron, $\sigma_T = (O, A, B, C)$. We set $\boldsymbol{A} = \vec{OA}$, with similar notations for B and C. As these tetrahedra are oriented, the integral over the whole shape is simply the sum of the integrals over them. Therefore,

$$\Omega_{nl}^m = \sum_{\sigma_T} \mathrm{sign}(V(\sigma_T)) \int_{\sigma_T} \overline{Z_{nl}^m(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{25}$$

The volume of the oriented tetrahedron is given by:

$$V(\sigma_T) = \frac{1}{6}\det(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \frac{1}{6}\begin{vmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ z_A & z_B & z_C \end{vmatrix}$$

3.2 Basic idea

Our task is then to evaluate the integrals of the Zernike polynomials over a tetrahedron $\sigma_T = (0, A, B, C)$. The first step is to perform a change of basis, parameterizing the tetrahedron in terms of the barycentric coordinates of the triangle T and the fractional distance to the origin (see figure 3).

A point $\boldsymbol{P}$ with spherical coordinates (r, θ, ψ) inside the tetrahedron can be characterized by its projection from the origin to the triangle T. Let $\boldsymbol{P_0}$ be this point. The spherical coordinates of this point are (r_0, θ, ϕ). Note that r_0 is a function of θ and ϕ. The integration over the tetrahedron proceeds in two steps, first a radial integration over r from 0 to r_0, and then an integration over the triangle only.

$$I_{nl}^m = \int_{\sigma_T} \overline{Z_{nl}^m(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} = \frac{3V(\sigma_T)}{S(T)} \int_T \frac{1}{r_0^3} \left(\int_0^{r_0} r^2 R_{nl}(r) dr \right) \overline{Y_l^m(\theta, \phi)} \mathrm{d}\boldsymbol{P_0}$$

This expression defines a basic 2-step process for evaluating those integrals and therefore the Zernike moments:

i) For a given $\boldsymbol{P_0}$ inside the triangle T with radius r_0, compute the radial integral

$$Q_{nl}(r_0) = \int_0^{r_0} r^2 R_{nl}(r) dr$$

ii) Integrate over all points $\boldsymbol{P_0}$ in the triangle T to get:

$$I_{nl}^m = \frac{3V(\sigma_T)}{S(T)} \int_T \frac{Q_{nl}(r_0)}{r_0^3} \overline{Y_l^m(\theta, \phi)} \mathrm{d}\boldsymbol{P_0}$$

In the following, we provide recurrence relationships to evaluate the Q_{nl} and describe a numerical way to evaluate exactly the integral in I_{nl}^m, using quadrature.

3.3 Computing the integrals Q_{nl}

We consider the different integrals $S_{nl}^k(r)$ of the radial functions $R_{nl}(r)$:

$$S_{nl}^k(r) = \int_0^r \rho^k R_{nl}(\rho) \mathrm{d}\rho$$

Prata and Rusch have derived relationships for similar integrals for the 2D Zernike radial polynomials [35]. We expand their results here. In appendix A, we show the following proposition that allows us to compute $S_{nl}^0(r)$:

Property 1. *For non negative integers n and l with $l \leq n-2$ and $n-l \equiv 0$ (mod 2), the following relationship hold:*

$$S_{nl}^0(r) = \frac{2l+3}{(2n+3)(l+1)} \Big(R_{n+1,l+1}(r) - R_{n-1,l+1}(r) \Big) - \frac{l+2}{l+1} S_{n,l+2}^0(r) \tag{26}$$

for $l = n$, we need additionally:

$$S_{nn}^0(r) = \frac{r^{n+1}}{n+1}$$

In appendix B, we then establish a recurrence on the $S_{nl}^k(r)$, in steps of 2 in k:

Property 2. *For non negative integers n and l with $l \leq n-2$ and $n-l \equiv 0$ (mod 2), and for positive k, the following relationship hold:*

$$K_1(n+2,l) S_{nl}^{k+2}(r) = -S_{n+2,l}^k() + K_2(n+2,l) S_{nl}^k(r) + K_3(n+2,l) S_{n-2,l}^k(r) \tag{27}$$

where K_1, K_2, and K_3 are defined in Equation 22.

The special case $k = 0$ leads to the following recurrence for the integral $Q_{nl}(r) = S_{nl}^2(r)$:

$$\begin{aligned} Q_{nl}(r) = {} & \frac{(n+2-l)(n+l+3)}{(2n+3)(2n+5)} S_{n+2,l}^0(r) \\ & + \frac{1}{2} \left(\frac{(2l+1)^2}{(2n+5)(2n+1)} + 1 \right) S_{nl}^0(r) + \frac{(n-l)(n+l+1)}{(2n+3)(2n+1)} S_{n-2,l}^0(r) \end{aligned} \tag{28}$$

for $l = n$, we need additionally:

$$Q_{nn}(r) = \frac{r^{n+3}}{n+3}$$

Finally, we show that for all positive integer n, and all positive integer l with $0 \leq l \leq n$ and n and l of the same parity, there exists a polynomial function U_{nl} of degree $n-l$ such that

$$Q_{nl}(r) = r^{l+3} U_{nl}(r) \tag{29}$$

The proof is simple. First, we notice that R_{nl} is written as :

$$R_{nl}(r) = r^l V_{nl}(r) \tag{30}$$

where V is a Jacobi polynomial (see equation 20) of degree $n-l$. Then,

$$Q_{nl}(r) = \int_0^r \rho^{l+2} V_{nl}(\rho) \mathrm{d}\rho \tag{31}$$

Using the expansion of V_{nl}, it is easy to show that $Q_{nl}(r)$ is a polynomial function of degree $n+3$, with r^{l+3} as a factor.

3.4 Computing the integrals I_{nl}^m

Recall that the triangle is defines as $T = (A, B, C)$ and that we need to compute over this triangle the integral

$$I_{nl}^m = \frac{3V(\sigma_T)}{S(T)} \int_T \frac{1}{r_0^3} Q_{nl}(r_0) \overline{Y_l^m(\theta, \phi)} \mathrm{d}\boldsymbol{P_0}$$

Let us define

$$g_{nl}^m(\boldsymbol{P_0}) = \frac{1}{r_0^3} Q_{nl}(r_0) \overline{Y_l^m(\theta, \phi)}. \tag{32}$$

Note that there is a function g for all triplets (n, l, m). We need to compute as exactly as possible,

$$\frac{1}{S(T)} \int_T g_{nl}^m(\boldsymbol{P_0}) \mathrm{d}\boldsymbol{P_0}$$

This integral can be approximated with a 2D Gaussian quadrature [36],

$$\frac{1}{S(T)} \int_T g_{nl}^m(\boldsymbol{P_0}) \mathrm{d}\boldsymbol{P_0} \approx \sum_{i=1}^{L} w_i g_{nl}^m(\alpha_i A + \beta_i B + \gamma_i C), \tag{33}$$

The sum is computed over N_p points on the triangle, with each point P_i defined by its barycentric coordinates $(\alpha_i, \beta_i, \gamma_i)$ with respect to (A, B, C), and w_i is a weight. The points and weights are said to define a quadrature rule. A rule is said to be of strength N if it is capable of exactly integrating any polynomial of maximal degree N over the domain (here the triangle). To apply such a scheme for our application, we need to consider two elements:

a) *Exactness.* As mentioned above, a 2D Gaussian quadrature may be exact if it is applied on a polynomial. This is our case. Indeed, recall that $Q_{nl}(r_0)$ is a polynomial function of degree $n+3$, with r_0^{l+3} as a factor. The function g can then be rewritten as

$$\begin{aligned} g_{nl}^m(\boldsymbol{x}) &= \frac{1}{r^3} Q_{nl}(r) \overline{Y_l^m(\theta, \phi)} \\ &= r^l U_{nl} \overline{Y_l^m(\theta, \phi)} \\ &= U_{nl} \overline{e_l^m(\boldsymbol{x})} \end{aligned}$$

where $e_l^m(\boldsymbol{x})$ are the harmonic polynomials (see equation 4). As U_{nl} is of degree $n-l$ and e_l^m is of degree l, the function g is a 2D polynomial function of degree n. A quadrature rule of strength n will therefore integrate g exactly.

b) *Number of points for the quadrature.* While it is well known that an n-point Gaussian rule is exact for all polynomials of degrees up to $2n-1$ in one dimension, the situation is more complex in higher dimensions. Xiao and Gimbutas [37] proposed an empirical rule for the minimal number of points N_p^* to integrate exactly a polynomial of order n:

$$N_p^* = \frac{(n+1)(n+2)}{6}$$

3.5 Two algorithms for computing Zernike moments

The previous subsections provide the elements for computing the contribution of one triangle of a surface mesh to the 3D Zernike moments of the shape enclosed with this mesh. We summarize those elements in Algorithm 1.

Algorithm 1 *Zernike moments associated with one triangle of a surface mesh*

procedure TRIANGLE(N, A, B, C, R)

Input: N: The maximum order for the 3D Zernike moments. A, B, C: The three vertices defining the triangle. R: The 2D Gauss quadrature rule

Initialize: $N(R)$ number of points in R. Initialize $I_{nl}^m = 0$. Compute V, the signed volume of the tetrahedron (O, A, B, C)

for $i = 1, \ldots, N(R)$ **do**

(1) Define (r_i, θ_i, ϕ_i) for point $\boldsymbol{P}_i$ in the quadrature rule.

(2) Evaluate $Q_{nl}(r_i)$ over all $n \in [0, N]$, l with $0 \leq l \leq n$ and n and l of same parity

(3) Evaluate $\overline{Y_l^m(\theta_i, \phi_i)}$ for all l with $0 \leq l \leq N$ and all m with $0 \leq m \leq l$

(4) Compute all $g_{nl}^m(\boldsymbol{P}_i)$ based on equation 32

(5) Update $I_{nl}^m = I_{nl}^m + 3V w_i g_{nl}^m(\boldsymbol{P}_i)$

end for

Output: The Zernike moments I_{nl}^m associated with triangle (A, B, C).

end procedure

Briefly, given a Gauss quadrature rule R defined by a set of weighted points $\boldsymbol{P}_i$, the algorithm proceeds by computing the functions $g_{nl}^m(\boldsymbol{P}_i)$ over all those weighted points, and then accumulating the results based on the quadrature rule given by equation 33. The functions $g_{nl}^m(\boldsymbol{P}_i)$ are computed from the $Q_{nl}^m(r_i)$ at r_i, the radial distance of $\boldsymbol{P}_i$, and from the $Y_l^m(\theta_i, \phi_i)$, at the inclination angle θ_i and azimuthal angle ϕ_i of $\boldsymbol{P}_i$. The corresponding procedure is defined as TRIANGLE. Details on its implementation are provided in the section below.

Given the procedure TRIANGLE, there are two possible algorithms that can then be used to compute the Zernike moments of a shape defined by a surface triangle mesh, one for an exact computation, and one for finite precision. We summarize them in algorithm 2 and algorithm 4, respectively.

Algorithm 2 *Exact Zernike moments for surface triangular meshes*

Input: The triangular mesh characterized by its number of vertices N_v and its number of facets M. The maximum order N for the 3D Zernike moments.

Initialize: Center and scale the mesh. Given N, define the 2D Gauss quadrature rule R to be applied on all triangles. Initialize $\Omega_{nl}^m = 0$

for $k = 1, \ldots, M$ **do**

(1) Define (A, B, C) the three vertices of triangle k.

(2) Compute $I_{nl}^m = \text{TRIANGLE}(N, A, B, C, R)$

(3) Update $\Omega_{nl}^m = \Omega_{nl}^m + I_{nl}^m$.

end for

Output: The exact Zernike moments Ω_{ln}^m of the shape.

As described on the previous subsection, the 2D Gauss quadrature rule of strength N will define exact Zernike moments of order up to N on any triangle of the surface mesh. For a given N, if a quadrature rule R with this strength exists, it is then sufficient to use the procedure TRIANGLE defined in algorithm 1 on all triangles of the surface mesh and then accumulate the results. As described below, we were able to generate quadrature rules for N up to 101.

The corresponding algorithm is of order $O(M \times N^5)$, where M is the number of triangles in the mesh and N the Zernike order. This can be derived as follows. The computation is performed independently on all triangles of the mesh, hence the factor M. For each facet, we need a quadrature rule R of strength N, which itself requires a number of points N_R that is empirically of order N^2 (see for example [37]). As N^3 functions g_{nl}^m need to be evaluated for each point in the quadrature rule, we get the overall time complexity $O(M \times N^5)$

Algorithm 3 *Approximate Zernike moments associated with one triangle with recursional splitting*

procedure TRIANGLESPLIT(N, A, B, C, R, S)

Input: N: The maximum order for the 3D Zernike moments. A, B, C: The three vertices defining the triangle. R: The 2D Gauss quadrature rule. S the number of times to split.

if $S = 0$ **then**

(1) Set I_{nl}^m = TRIANGLE(N, A, B, C, R)

else

(2) Compute the middles A', B', C' of segments BC, CA, AB

(3) Set J_{nl}^m = TRIANGLESPLIT$(N, A, C', B', R, S-1)$

(4) Set K_{nl}^m = TRIANGLESPLIT$(N, B, A', C', R, S-1)$

(5) Set L_{nl}^m = TRIANGLESPLIT$(N, C, B', A', R, S-1)$

(6) Set M_{nl}^m = TRIANGLESPLIT$(N, A', B', C', R, S-1)$

(7) Set $I_{nl}^m = J_{nl}^m + K_{nl}^m + L_{nl}^m + M_{nl}^m$

end if

Output: The Zernike moments I_{nl}^m associated with triangle (A, B, C).

end procedure

While algorithm 2 is deemed exact, it suffers from two major limitations. First, it requires quadrature rules with large strengths. Second, such rules include large number of sampling points, leading to an overall time complexity $O(M \times N^5)$. Quadrature rules, however, are known to converge fast. As such, it is often not necessary to go to the maximum strength that is required for an exact computation. If we are willing to accept a finite precision, a more efficient procedure can be derived, as described in algorithm 4. Briefly, the Zernike moments I_{nl}^m over a triangle T are evaluated over quadrature rules of increasing strengths. When the difference between those Zernike moments computed over two successive rules falls below a tolerance, the quadrature is deemed to have converged and the computation stops. If the strongest quadrature rule (here 101) is not enough to reach the desired tolerance, we repeatedly split the triangle in four equal triangles and we sum the results.

We will show that in general, this method converge significantly faster than the exact algorithm. It is not possible, however, to provide an expected time complexity for this method with finite precision as the order of the approximation used depends on N and on each facet, in particular its size.

4 Reconstructing a shape from its 3D Zernike moments

Recall that once a shape S has been characterized with its Zernike moments Ω_{nl}^m, its density $\rho_S(\boldsymbol{x})$ at any point $\boldsymbol{x}$ in $\mathbb{R}^3$ can be reconstructed using equation 13

$$\begin{aligned} \rho_S(\boldsymbol{x}) &= \sum_n^N \sum_l \sum_{m=-l}^l \Omega_{nl}^m Z_{nl}^m(\boldsymbol{x}) \\ &= \sum_n^N \sum_l \sum_{m=-l}^l \Omega_{nl}^m \sqrt{2n+3} R_{nl}(r) Y_l^m(\theta, \phi) \end{aligned} \tag{34}$$

Algorithm 4 *Approximate Zernike moments for surface triangular meshes*

Input: The triangular mesh characterized by its number of vertices N_v and its number of facets M. The maximum order N for the 3D Zernike moments. L the list of succesive quadratures to try. TOL: tolerance for convergence

Initialize: Center and scale the mesh. Initialize $\Omega_{nl}^m = 0$

for $k = 1, \ldots, M$ **do**

(1) Define (A, B, C) the three vertices of triangle k.

(2) Initialize $I_{nl}^m(\text{old}) = 0$

for $R \in L$ **do**

(3) Compute $I_{nl}^m(R) = \text{TRIANGLE}(N, A, B, C, R)$

(4) Compute err $= ||I_{nl}^m(R) - I_{nl}^m(\text{old})||$

(5) If err $<$ TOL, break

(6) Set $I_{nl}^m(\text{old}) = I_{nl}^m(R)$

end for

if err $<$ TOL **then**

(7) Set $I_{nl}^m = I_{nl}^m(R)$.

else

for $S = 1, \ldots$ **do**

(8) Set $I_{nl}^m(S) = \text{TRIANGLESPLIT}(N, A, B, C, R, S)$

(9) Compute err $= ||I_{nl}^m(S) - I_{nl}^m(\text{old})||$

(10) If err $<$ TOL, break

(11) Set $I_{nl}^m(\text{old}) = I_{nl}^m(S)$

end for

(12) Set $I_{nl}^m = I_{nl}^m(S)$.

end if

(13) Update $\Omega_{nl}^m = \Omega_{nl}^m + I_{nl}^m$.

end for

Output: The Zernike moments Ω_{nl}^m of the shape.

where (r, θ, ϕ) are the spherical coordinates of $\boldsymbol{x}$. The reconstruction is exact when $N \to +\infty$. While the Zernike moments Ω_{nl}^m and the Zernike polynomials are complex, the reconstruction $\rho_S(\boldsymbol{x})$ is real. Indeed, recall that for an $m*$ negative,

$$\begin{aligned} \Omega_{nl}^{m*} &= (-1)^{m*}\overline{\Omega_{nl}^{-m*}} \\ Y_l^{m*} &= (-1)^{m*}\overline{Y_l^{-m*}} \end{aligned}$$

and that $Y_l^0(\theta, \phi) = N_l^m P_l^m(\cos(\theta))$ is real. Then,

$$\begin{aligned} \rho_S(\boldsymbol{x}) &= \sum_n^N \sqrt{2n+3} \sum_l R_{nl}(r) \left(\Omega_{nl}^0 Y_l^0(\theta, \phi) + \sum_{m=1}^l (\Omega_{nl}^m Y_l^m(\theta, \phi) + \overline{\Omega_{nl}^m Y_l^m(\theta, \phi)}) \right) \\ &= \sum_n^N \sqrt{2n+3} \sum_l R_{nl}(r) \left(\Omega_{nl}^0 Y_l^0(\theta, \phi) + 2 \sum_{m=1}^l \Re(\Omega_{nl}^m Y_l^m(\theta, \phi)) \right) \end{aligned}$$

where $\Re$ indicates the real part.

The equation above defines a simple algorithm for reconstructing the shape density in $\mathbb{R}^3$. If the points $\boldsymbol{x}$ are chosen to be the nodes of a 3D grid, a surface mesh can then be reconstructed using the marching tetrahedron algorithm [38]. We note, however, that special care is needed

when evaluating the radial polynomials $R_{nl}(r)$ and the spherical harmonics $Y_l^m(\theta, \phi)$ to avoid numerical instabilities when n is large. This is described below in the Implementation section.

5 Implementation

The computation of the Zernike moments of a shape described by a surface triangular mesh is performed either with the exact algorithm 2 or with the finite precision algorithm 4. Both algorithms rely heavily on the function that computes the geometric moment associated with a triangle of the mesh, TRIANGLE (algorithm 1). We identify four elements that are essential for the implementations of those algorithms:

1) Definitions of the quadrature rules for integration on a triangle,

2) Efficient computations of $R_{nl}(r)$ and $Q_{nl}(r)$,

3) Efficient computations of the spherical harmonics $Y_l^m(\theta, \phi)$

4) Efficient computations of g_{nl}^m

We describe the specifics for those elements.

5.1 Quadrature rules for integration over a triangle

A quadrature rule over a domain D is defined through its ability to integrate exactly over D the set of basis polynomials of degrees $n \leq N$, $\mathbb{P}^N$. This set has an infinite number of representations. The simplest of those representations is to consider monomials. In two dimensions, those monomials are $\{x^i y^j, i+j \leq N\}$. Unfortunately, monomials of high degrees are extremely sensitive to small perturbations. This gives rise to systems which are poorly conditioned and hence difficult to solve numerically [39]. We have used instead the approach of Witherden and Vincent [40] to derive our quadrature rules. They proposed to use orthogonal polynomials ψ_{ij} as a basis of $\mathbb{P}^N$ with $i + j \leq N$, such that

$$\int_D \psi_{ij}(\boldsymbol{x})\psi_{kl}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \delta_{ik}\delta_{jl}$$

where δ is the Kronecker delta. By taking $\psi_{00}(\boldsymbol{x}) = 1/c$, they define the error associated with a quadrature rule with N_p points as:

$$\chi^2(N) = \sum_{ij} \left\{ \sum_k^{N_p} w_k \psi_{ij}(\boldsymbol{x}_k) d\boldsymbol{x} - c\delta_{i0}\delta_{j0} \right\}^2 \tag{35}$$

In Witherden and Vincent's schemes, constructing an N_p rule of strength N is then akin with finding a set of points $\boldsymbol{x}_k$ and associated weights w_k that minimize $\chi^2(N)$. They provided an open source software package, PolyQuad (available at `https://github.com/PyFR/Polyquad`) for this task. We have run PolyQuad for strengths between 3 and 101 to generate the quadrature rules that we have used for computing Zernike moments. In appendix C, we list the number of points required for each strength. The actual number of points is found to be similar to the empirical bound of $(N + 1)(N + 2)/6$ proposed by Xiao and colleagues [37].

5.2 Efficient computations of the polynomials $R_{nl}(r)$ and $Q_{nl}(r)$

As described in section 2, it is crucial to compute the radial polynomials $R_{nl}(r)$ accurately. A naive computation using its monomial decomposition does not achieve this accuracy for high order n. Instead, we have used the recurrence 21 that is derived from the properties of Jacobi polynomials (see section 2).

The polynomial $Q_{nl}(r)$ are the indefinite integrals of the radial polynomials $R_{nl}(r)$ weighted by r^2. Properties 1 and 2 (major results of this paper) provide simple recurrence for computing those integrals.

5.3 Efficient computations of the spherical harmonics $Y_l^m(\theta, \phi)$

The spherical harmonics are related to the Legendre polynomials, from which they inherit many properties. In particular, they can be computed recursively. We have used the following recurrences:

Property 3. *For non negative integers l and m with $0 \leq m \leq l$ the following relationships hold:*

i) For $l > 1$ and $0 \leq m < l - 1$:

$$Y_l^m(\theta, \phi) = \sqrt{\frac{(2l+1)(2l-1)}{(l+m)(l-m)}} \cos\theta Y_{l-1}^m(\theta, \phi) - \sqrt{\frac{(2l+1)(l+m-1)(l-m-1)}{(2l+3)(l+m)(l-m)}} Y_{l-2}^m(\theta, \phi) \quad (36)$$

ii) For $l > 0$ and $m = l - 1$:

$$Y_l^{l-1}(\theta, \phi) = \sqrt{2l+1} \cos\theta Y_{l-1}^{l-1}(\theta, \phi) \quad (37)$$

iii) For $l > 0$ and $m = l$:

$$Y_l^l(\theta, \phi) = -\sin\theta \sqrt{\frac{2l+1}{2l}} e^{i\phi} Y_{l-1}^{l-1}(\theta, \phi) \quad (38)$$

With the initialization $Y_0^0(\theta, \phi) = 1/\sqrt{4\pi}$, equations 36 to 38 provide an efficient scheme for computing all spherical harmonics at angles (θ, ϕ).

For sake of completeness, we provide a proof of this property in Appendix D.

5.4 Efficient computation of the product g_{nl}^m

At step 4 of the TRIANGLE procedure, we must form the product given in Equation 32. As we have managed to compute Q_{nl} and Y_l^n in time $O(N^2)$, this is where most of the computation time will go, since there are of order N^3 terms to compute. To streamline this very simple computation (it is a simple product), we have optimized the ordering of the terms.

Remember that n and l must have the same parity, one may then order the Q_{nl} terms (and likewise for R_{nl}) like this (l varying first and n varying last) Q_{00}, Q_{11}, Q_{20}, Q_{22}, Q_{31}, Q_{33}, Q_{40}, Q_{42} and so on. The problem here is that the l are out of order which is bad for computing the g_{nl}^m efficiently.

We choose this alternative ordering: we interleave the successive even and odd values of n: Q_{00}, Q_{11}, Q_{20}, Q_{31}, Q_{22}, Q_{33}, Q_{40}, Q_{51} and so on. Now the l are in order, we no longer separate the even and odd l values.

6 Numerical results

6.1 Testing accuracy

In this article, we propose two new algorithms to accurately compute 3D Zernike moments for triangular surface meshes. To sustain our claim of accuracy, we need tools to measure this accuracy. We propose three such tools.

To build them, we define the signatures of a family of Zernike moments by

$$\sigma_n\{\Omega\} = \sum_{l=0}^{n} \sum_{m=-l}^{l} |\Omega_{nl}^m|^2. \quad (39)$$

Again the sum on l runs only for l and n with the same parity. It is easy to check that the σ_n are invariant by rotation. These signatures allow to summarize information for each value of n.

The first test ("against exact") is a comparison of the computed moments with exact results. Of course this test is limited to very particular shapes for which we are able to compute exact results to high order. For this test we chose a cube with vertices of coordinates $(\pm 1/\sqrt{3}, \pm 1/\sqrt{3}, \pm 1/\sqrt{3})$. To compute the Zernike moments exactly, we use a symbolic algebra system (namely Mathematica) to exactly compute the integrals of Eq. 11 up to $n = 100$ (this took 3 full days of computation running in parallel on a 4 core every day computer). We thus obtained the Zernike moments of the cube to machine precision. We thus have our first accuracy test

$$\alpha_n = \sigma_n\{\Omega(\text{computed}) - \Omega(\text{exact})\}. \tag{40}$$

This test scales like ϵ^2 where ϵ is the error made on the computed moments.

The second test ("against self") is an auto coherent test. It compares rotation invariants computed from the Zernike moments, for two different orientations of the same shape. This gives us our second accuracy test

$$\beta_n = |\sigma_n\{\Omega(\text{original})\} - \sigma_n\{\Omega(\text{rotated})\}|. \tag{41}$$

This test scales like ϵ.

The third test ("reconstruction") is a visual test. We compute the Zernike moments of the shape and we reconstruct a new shape from these moments, and we visually compare the two. For this, we use the regularized marching tetrahedra algorithm as described in [41]. Note that for the reconstruction, accuracy is also an issue and it is important to evaluate the Zernike polynomials using the same procedure as before.

6.2 Results of the tests

We will compare three algorithms: (i) the Pozo et al. algorithm [30] modified by Koehl [31]. We call it the PK algorithm. It is the state of the art of the algorithms using the geometric moments (ii) our exact algorithm presented above, (iii) our approximate algorithm, also presented above, the requested global precision was set to 10^{-6} divided by the number of facets.

In Fig. 4, we show the results of the tests against exact using maximum order required $N = 100$. Only even order Zernike moments are shown as the others are null. In this case, the approximate algorithm reaches the maximum triangle quadrature order $n = 101$ which is exact for all facets and thus gives the same results as the exact algorithm. We clearly see that both our algorithms perform perfectly, whereas the PK algorithm steadily loses precision as n increases, it becomes meaningless around $n = 45$.

For the next two tests, we use two triangulated objects as shown of figure 5. We call the first one "rings" and the second one "man".

The results for the tests against self are shown on Figure 6 for "rings" and on Figure 7 for "man". We used $N = 100$ for the three algorithms. Again, we see that the PK algorithm completely loses precision around $N = 45$ whereas both our algorithms stay stable. Note how our approximate algorithm usually achieves a much better precision than requested (here we asked for $p = 10^{-6}$, but obtained around 10^{-12}).

Finally, examples of reconstructions are shown on Figure 8 and 9 for the approximate algorithm. Similar results are found for the exact algorithm in the range it can cover. Note how the finest details are visible at $N = 300$: the ripples on object "rings" present on the original, or the face and foot details on "man".

6.3 Speed

For the first algorithm, we have run some speed tests varying the objects and the requested N. Our measures are coherent with a computation time roughly equal to $\tau M N^5$. On our computer (an intel core i5-3470 at 3.2 GHz), we get $\tau \approx 30$ps. There are fluctuations to this computation time depending whether the chosen integration schemes fits closely or not.

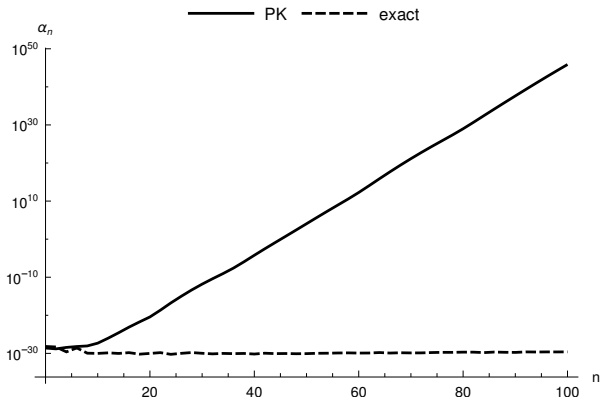

Figure 4: Test against exact for the three algorithms. The approximate algorithm is indistinguishable from the exact one in this case and is thus not shown.

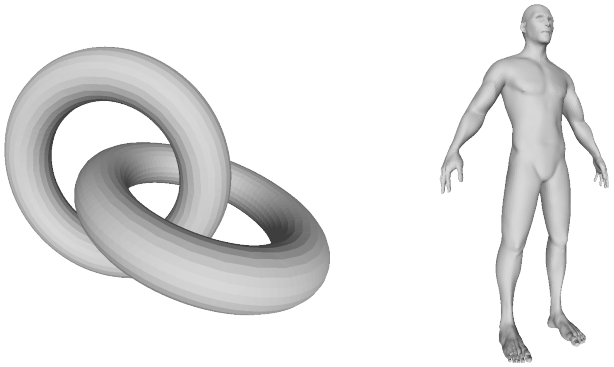

Figure 5: Our test objects "rings" and "man", with respectively $M = 9792$ and $M = 48918$ facets.

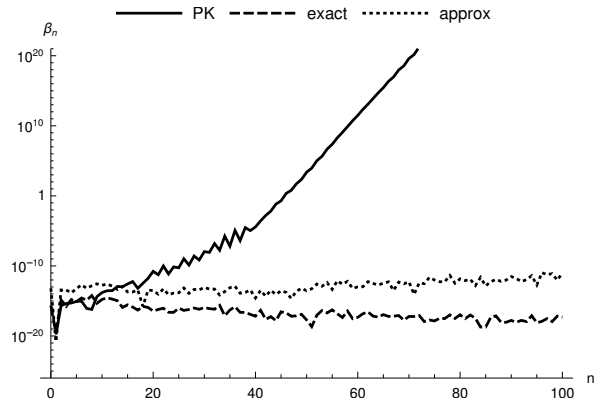

Figure 6: Test against self for the three algorithms for object "rings".

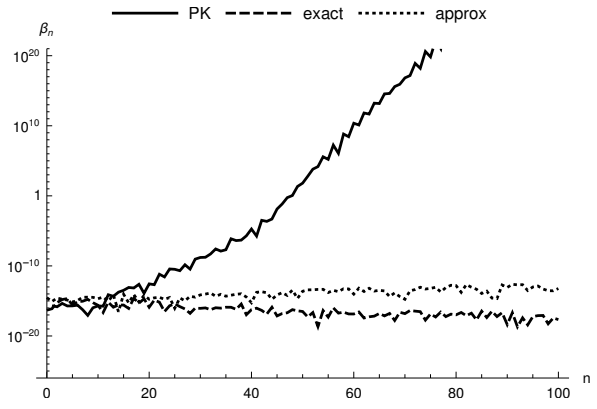

Figure 7: Test against self for the three algorithms for object "man".

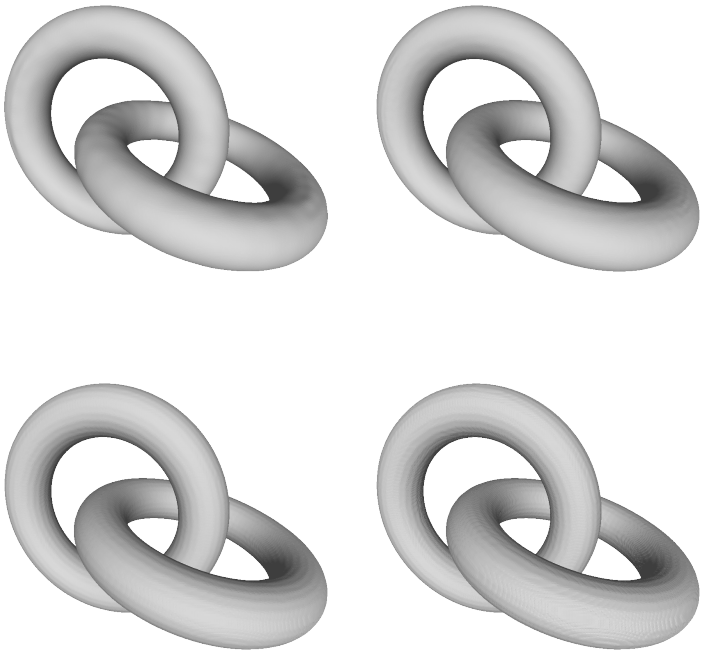

Figure 8: Reconstruction test for the approximate algorithm for object "rings". The pictures corresponds to $N = 50$, $N = 100$, $N = 200$, $N = 300$ from left to right and top to bottom.

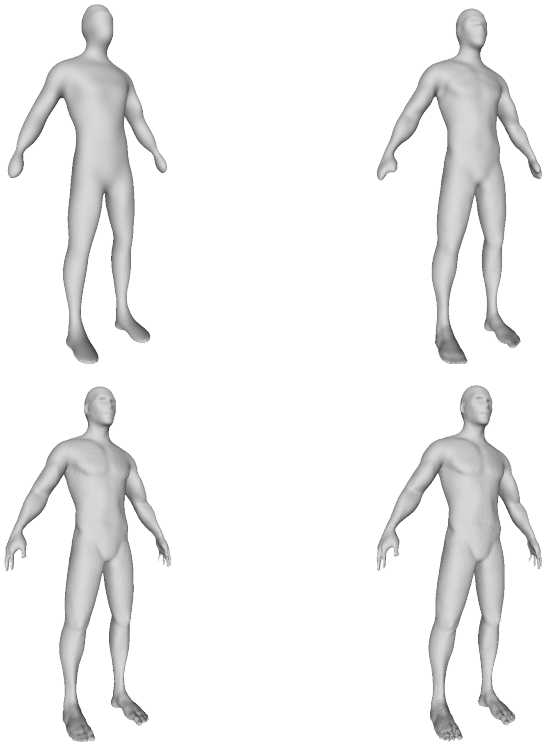

Figure 9: Reconstruction test for the approximate algorithm for object "man". The pictures corresponds to $N = 50$, $N = 100$, $N = 200$, $N = 300$ from left to right and top to bottom.

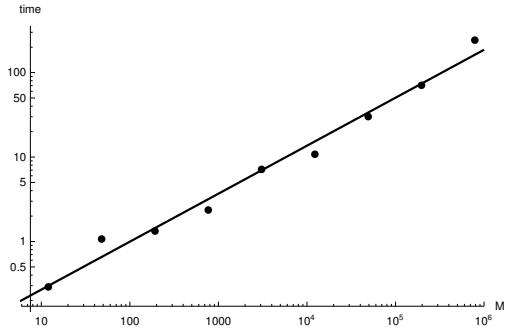

Figure 10: Computation time (in seconds) of the approximate algorithm with $N = 50$ on a cube with different number of facets. The line has equation $0.073M^{0.568}$.

For the second algorithm, we have tested the M dependence, because the larger the number of facets, the smaller the facets and the lower we need to go in the approximation ladder. To do this, we took the cube and repeatedly split each facets in four, at each steps we ran our approximate algorithm with $N = 50$ and precision $p = 10^{-6}$ divided by the number of facets. The results are presented on Figure 10. These measurements give a size dependence in M^{μ} with $\mu \approx 0.57$ that is approximately $\sqrt{M}$.

The precision dependence of the computation time is small but discontinuous, since the convergence of the procedure is very fast. Thus the computation time will stay constant on large domains of precision and will suddenly jump by a factor 2, when the algorithm needs to go one step further.

Globally the computation time of the approximate algorithm is very roughly consistent with $\tau\sqrt{M}N^5$. On our computer, we have $\tau \approx 600$ps (but it fluctuates quite a lot). This means that for a typical objects with 10 000 facets the approximate algorithm is roughly 5 times faster than the exact one, and the larger the number of facets the larger the gain in computation time.

7 Conclusion

We have introduced two stable algorithms to accurately compute 3D Zernike moments for objects defined by a triangular mesh surface. In opposition, to the previous state of the art algorithm which typically completely loses precision around order $N = 45$, both algorithms do not lose precision with growing order.

The first algorithm is exact and works up to $N = 101$ with complexity $O(MN^5)$. It is limited by the availability of quadrature rule on the triangle with sufficient order of exactness.

The second algorithm is approximate but can compute the moments to a given precision. It is significantly faster than the exact algorithm with an effective complexity of roughly $O(\sqrt{M}N^5)$. This algorithm has no limitation in the maximum order N except the duration of the computation. Here we have presented results up to $N = 300$. We believe that this second algorithm is nearly always preferable to the first.

We have shown several tests of accuracy for our algorithm, a test against exact results, an auto coherent test and a reconstruction test.

The free software implementing these algorithms is available at `https://github.com/jerhoud/zernike3d`

References

[1] M.L. Zelditch, D.L. Swiderski, and H.D. Sheets. *Geometric Morphometrics for Biologists. A Primer*. Elsevier: Academic Press, London, 2012.

[2] H. Peng. Bioimage informatics: a new area of engineering biology. *Bioinformatics*, 24:1827–1836, 2008.

[3] K. Khairy and J. Howard. Spherical harmonics-based parametric deconvolution of 3D surface images using bending energy minimizations. *Med. Image Anal.*, 12:217–227, 2008.

[4] L. Shamir, J.D. Delaney, N. Orlov, D.M. Eckley, and I.G. Goldberg. Pattern recognition software and techniques for biological image analysis. *PLoS Comput. Biol.*, 6:e1000974, 2010.

[5] D. Toomre and J. Bewersdof. A new wave of cellular imaging. *Annu. Rev. Cell. Dev. Biol.*, 26:285–314, 2010.

[6] D.W. Thompson. *On growth and form*. University Press, Cambridge, 1917.

[7] M. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Infor. Theory*, 8:179–187, 1962.

[8] M. Teague. Image analysis via the general theory of moments. *J. Opt. Soc. Amer.*, 70:920–930, 1980.

[9] C.H. Teh and R.T. Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Machine Intell.*, 10:496–513, 1988.

[10] R.J. Prokop and A.P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *Graphical models and image processing*, 54:438–460, 1992.

[11] E.W. Hobson. *The Theory of Spherical and Ellipsoidal Harmonics*. Chelsea Co., New York, NY, 1955.

[12] W. E. Byerly. An elementary treatise on fourier's series, and spherical, cylindrical, and ellipsoidal harmonics, with applications to problems in mathematical physics. In *Spherical Harmonics*, pages 195–218, New York, NY, 1959. Dover.

[13] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Eurographics Symp. Geometric Proc.*, pages 156–164, 2003.

[14] Anna Medyukhina, Marco Blickensdorf, Zoltán Cseresnyés, Nora Ruef, Jens V Stein, and Marc Thilo Figge. Dynamic spherical harmonics approach for shape classification of migrating cells. *Scientific reports*, 10:1–12, 2020.

[15] F. Zernike. Beugungstheorie des Schneidenver-fahrens und seiner verbesserten Form, der Phasenkontrastmethode. *Physica*, 1(7):689–704, 1934.

[16] Pablo Toharia, Oscar D Robles, Ángel Rodríguez, and Luis Pastor. A study of zernike invariants for content-based image retrieval. In *Pacific-Rim Symposium on Image and Video Technology*, pages 944–957. Springer, 2007.

[17] N. Canterakis. 3D Zernike moments and Zernike affine invariants for 3D image analysis and recognition. In *Proc. 11th Scandinavian Conf. Image Anal.*, pages 85–93, 1997.

[18] M. Novotni and R. Klein. 3D Zernike descriptors for content based shape retrieval. In *Proc. ACM symposium on solid and physical modeling*, pages 216–225, 2003.

[19] M. Novotni and R. Klein. Shape retrieval using 3d zernike descriptors. *Computer Aided Design*, 36:1047–1062, 2004.

[20] K. Wang, T. Zhu, Y. Gao, and J. Wang. Efficient terrain matching with 3-D Zernike moments. *IEEE Trans. Aerosp. Elec. Sys.*, 55:226–235, 2018.

[21] Bing Ma, Ye Zhang, and Shu Tian. Building reconstruction using three-dimensional zernike moments in digital surface model. In *2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1324–1327. IEEE, 2018.

[22] Lee Sael, Bin Li, David La, Yi Fang, Karthik Ramani, Raif Rustamov, and Daisuke Kihara. Fast protein tertiary structure retrieval based on global surface shape similarity. *Proteins: Struct. Func. Bioinfo.*, 72:1259–1273, 2008.

[23] V. Venkatraman, Y. Yang, L. Sael, and D. Kihara. Protein-protein docking using region-based 3D Zernike descriptors. *BMC Bioinformatics*, 10:407, 2009.

[24] S. Daberdaku and C. Ferrari. Exploring the potential of 3D Zernike descriptors and svm for protein–protein interface prediction. *BMC Bioinformatics*, 19:35, 2018.

[25] S. Daberdaku and C. Ferrari. Antibody interface prediction with 3D Zernike descriptors and svm. *Bioinformatics*, 35:1870–1876, 2019.

[26] L. Di Rienzo, E. Milanetti, J. Alba, and M. D'Abramo. Quantitative characterization of binding pockets and binding complementarity by means of zernike descriptors. *J. Chem. Infor. Model.*, 60:1390–1398, 2020.

[27] K.M. Hosny and M.A. Hafez. An algorithm for fast computation of 3d zernike moments for volumetric images. *Math. Probl. Eng.*, 2012, 2012.

[28] Daniel Berjón, Sergio Arnaldo, and Francisco Morán. A parallel implementation of 3d zernike moment analysis. In *Parallel Processing for Imaging Applications*, volume 7872, pages 83–89. SPIE, 2011.

[29] S. Lien and J.T. Kajiya. Symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. *IEEE Comput. Graph. Appl.*, 4:35–41, 1984.

[30] J.M. Pozo, M.-C. Villa-Uriol, and A.F. Frangi. Efficient 3D geometric and Zernike moments computation from unstructured surface meshes. *IEEE Trans. Pattern Anal. Machine Intell.*, 33:471–484, 2011.

[31] P. Koehl. Fast recursive computation of 3d geometric moments from surface meshes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34:2158–2163, 2012.

[32] A.-W. Deng and C.-Y. Gwo. A stable algorithm computing high-order 3d zernike moments and shape reconstructions. In *Proceedings of the 2020 4th International Conference on Digital Signal Processing*, ICDSP 2020, page 38–42, New York, NY, USA, 2020. Association for Computing Machinery.

[33] Richard J Mathar. Zernike basis to cartesian transformations. *arXiv preprint arXiv:0809.2368*, 2008.

[34] A.J.E.M Janssen. Generalized 3D Zernike functions for analytic construction of band-limited line-detecting wavelets. *arXiv preprint arXiv:1510.04837*, 2015.

[35] Aluizio Prata and WVT Rusch. Algorithm for computation of zernike polynomials expansion coefficients. *Applied Optics*, 28:749–754, 1989.

[36] A.H. Stroud. *Approximate calculation of multiple integrals*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[37] H. Xiao and Z. Gimbutas. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Comput. Math. with Appl.*, 59:663–676, 2010.

[38] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Trans. Infor. Sys.*, 74:214–224, 1991.

[39] Linbo Zhang, Tao Cui, and Hui Liu. A set of symmetric quadrature rules on triangles and tetrahedra. *J. Comput. Math.*, pages 89–96, 2009.

[40] F.D. Witherden and P.E. Vincent. On the identification of symmetric quadrature rules for finite element methods. *Comput. Math. with Appl.*, 69:1232–1241, 2015.

[41] G.M. Treece, R.W. Prager, and A.H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Comput. Graph.*, 23:583–598, 1999.

[42] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.1.6 of 2022-06-30. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

Appendix A: A recurrence for $S_{nl}(0,r)$

We start with an integral representation of the 3D Zernike radial polynomials [34]:

$$R_{nl}(\rho) = \frac{2}{\pi}(-1)^{\frac{n-l}{2}} \int_0^{+\infty} j_{n+1}(q) j_l(\rho q) q \mathrm{d}q \tag{A1}$$

where j_k are spherical Bessel functions. The derivative of $R_{nl}(\rho)$ with respect the ρ is then:

$$\frac{dR_{nl}}{d\rho}(\rho) = \frac{2}{\pi}(-1)^{\frac{n-l}{2}} \int_0^{+\infty} j_{n+1}(q) \frac{\partial j_l(\rho q)}{\partial \rho} q \mathrm{d}q \tag{A2}$$

Using the following relationship for spherical Bessel functions ([42], equation 10.51.1):

$$q j_{n+1}(q) = (2n+1) j_n(q) - q j_{n-1}(q))$$

we get:

$$\begin{aligned}
\frac{dR_{nl}}{d\rho}(\rho) &= \frac{2}{\pi}(-1)^{\frac{n-l}{2}} \int_0^{+\infty} q j_{n+1}(q) \frac{\partial j_l(\rho q)}{\partial(\rho q)} q \mathrm{d}q \\
&= \frac{2}{\pi}(-1)^{\frac{n-l}{2}} (2n+1) \int_0^{+\infty} j_n(q) \frac{\partial j_l(\rho q)}{\partial(\rho q)} q \mathrm{d}q - \frac{2}{\pi}(-1)^{\frac{n-l}{2}} \int_0^{+\infty} q j_{n-1}(q) \frac{\partial j_l(\rho q)}{\partial(\rho q)} q \mathrm{d}q \\
&= \frac{2}{\pi}(-1)^{\frac{n-l}{2}} (2n+1) \int_0^{+\infty} j_n(q) \frac{\partial j_l(\rho q)}{\partial(\rho q)} q \mathrm{d}q + \frac{dR_{n+2,l}}{d\rho}(\rho)
\end{aligned} \tag{A3}$$

We use now the following relationship for spherical Bessel functions ([42], equation 10.51.1):

$$\frac{dj_l(x)}{dx} = \frac{l}{2l+1} j_{l-1}(x) - \frac{l+1}{2l+1} j_{l+1}(x) \tag{A4}$$

to get:

$$\begin{aligned}
\frac{dR_{nl}}{d\rho}(\rho) = \frac{2}{\pi}(-1)^{\frac{n-l}{2}} (2n+1) \int_0^{+\infty} j_n(q) \left(\frac{l}{2l+1} j_{l-1}(x) - \frac{l+1}{2l+1} j_{l+1}(x) \right) q \mathrm{d}q \\
+ \frac{dR_{n+2,l}}{d\rho}(\rho)
\end{aligned} \tag{A5}$$

This equation leads to,

$$\frac{dR_{nl}}{d\rho}(\rho) = \frac{(2n+1)l}{2l+1} R_{n-1,l-1}(\rho) + \frac{(2n+1)(l+1)}{2l+1} R_{n-1,l+1}(\rho) + \frac{dR_{n+2,l}}{d\rho}(\rho)$$

After integration over $[0, r]$,

$$R_{nl}(r) = \frac{(2n+1)l}{2l+1} S_{n-1,l-1}(0,r) + \frac{(2n+1)(l+1)}{2l+1} S_{n-1,l+1}(0,r) + R_{n+2,l}(r) \tag{A6}$$

which concludes the proof of Equation 26, the recurrence over the $S_{nl}(0,r)$.

The initialization follows from

$$S_{n,n}(0,r) = \int_0^r R_{n,n}(\rho) \mathrm{d}\rho = \int_0^r \rho^n \mathrm{d}\rho = \frac{r^{n+1}}{n+1} \tag{A7}$$

Appendix B: A recurrence for $S_{nl}(k,r)$

We start from the recurrence over the $R_{nl}(r)$ (21 in the main body of the text):

$$R_{nl}(\rho) = K_1(n,l)\rho^2 R_{n-2,l}(\rho) + K_2(n,l)R_{n-2,l}(\rho) + K_3(n,l)R_{n-4,l}(\rho) \tag{B1}$$

where K_1, K_2, and K_3 were defined in equation 22 in the main text. Let k be a positive integer. After multiplication with ρ^k, we get:

$$\rho^k R_{nl}(\rho) = K_1(n,l)\rho^{k+2} R_{n-2,l}(\rho) + \rho^k K_2(n,l)R_{n-2,l}(\rho) + \rho^k K_3(n,l)R_{n-4,l}(\rho) \tag{B2}$$

which we integrate over $[0,r]$,

$$S_{nl}(k,r) = K_1(n,l)S_{n-2,l}(k+2,r) + K_2(n,l)S_{n-2,l}(k,r) + K_3(n,l)S_{n-4,l}(k,r) \tag{B3}$$

Shifting $n \leftarrow n-2$, we get:

$$\begin{aligned} K_1(n+2,l)S_{nl}(k,r) = S_{n+2,l}(k+2,r) - K_2(n+2,l)S_{nl}(k,r) \\ - K_3(n+2,l)S_{n-2,l}(k,r) \end{aligned} \tag{B4}$$

This then yields equation 28.

Starting with $S_{nl}(0,r)$, repeated use of equation B4 allows us to compute all $S_{nl}(k,r)$ for k even. While this is enough for this paper, for sake of completeness, we show how the same integrals can be evaluated for k odd.

The recurrence on R_{nl} expressed in Equation 21 has a coefficient with ρ to the power 2, leading to the even recurrence. Janssen in his work on generalized Zernike functions derived a different recurrence on R_{nl} (his equation 80):

$$R_{n+l,l}(\rho) = \frac{2n+3}{2n+2}\rho\left(\frac{2l+2}{2l+1}R_{n,l+1}(\rho) + \frac{2l}{2l+1}R_{n,l-1}(\rho)\right) - \frac{n+2}{n+1}R_{n-1,l}(\rho) \tag{B5}$$

Applications of the recurrence require the initialisation $R_{0,0}(\rho) = 1$, and setting $R_{nl} \equiv 0$ when $n < l$. After integration over $[0,r]$, we get

$$S_{n+l,l}(0,\rho) = \frac{2n+3}{2n+2}\left(\frac{2l+2}{2l+1}S_{n,l+1}(1,\rho) + \frac{2l}{2l+1}S_{n,l-1}(1,\rho)\right) - \frac{n+2}{n+1}S_{n-1,l}(0,\rho) \tag{B6}$$

which we rewrite as

$$\begin{aligned} S_{n,l-1}(1,\rho) = -\frac{l+1}{l}S_{n,l+1}(1,\rho) + \frac{(2n+2)(2l+1)}{(2n+3)(2l)}S_{n+1,l}(0,\rho) \\ + \frac{(2n+4)(2l+1)}{(2n+3)(2l)}S_{n-1,l}(0,\rho) \end{aligned} \tag{B7}$$

Equation B7 provides a recurrence for computing $S_{nl}(1,r)$ from $S_{nl}(0,r)$. Integrals $S_{nl}(k,r)$ with k odd, $k > 1$ can then be derived by repeated use of equation B4, starting with $S_{nl}(1,r)$.

Appendix C: Characteristics of the triangle quadrature rules used in this work

Strength N	#points N_p	bound, B	E
*3	4	4	0.83
*5	7	7	1
*7	13	12	1
9	19	19	0.96
*11	28	26	1
13	37	35	0.94
*17	60	57	0.95
21	87	85	0.97
*25	120	117	0.97
31	181	176	0.97
*37	255	247	0.97
43	348	330	0.95
*51	501	460	0.92
65	814	737	0.90
*73	1030	925	0.90
81	1263	1135	0.90
*101	2007	1751	0.87

Table 1: Characteristics of the triangle quadrature rules used in our algorithms. All those rules were constructed with the program PolyQuad (`https://github.com/PyFR/Polyquad`) by Witherden and Vincent [40]. Stars near strengths identify the rules that are used in the finite precision algorithm. The bound B (third column) is the empirically proposed bound on the number of points, $B = \lceil (N+1)(N+2)/6 \rceil$, as proposed by Xiao and Gimbutas [37]. E (fourth column) is the "efficiency", defined as: $E = (N+1)(N+2)/(6 * N_p)$. Efficient quadrature rules have E close to 1.

Appendix D: A recurrence for $Y_l^m(\theta, \phi)$

We will prove property 3 that establishes a recurrence on the spherical harmonics. Recall that

$$Y_l^m(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi} \tag{D1}$$

where P_l^m are the associated Legendre polynomials. We will look at three cases:

i) $l > 1$ and $m < l - 1$

For those values of l and m, the associated Legendre polynomials follow the simple recurrence

$$(l-m)P_l^m(x) = (2l-1)xP_{l-1}^m(x) - (l+m-1)P_{l-2}^m(x) \tag{D2}$$

Replacing equation D2 into D1, we get,

$$\begin{aligned} Y_l^m(\theta,\phi) &= \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} e^{im\phi} \left(\frac{2l-1}{l-m}\cos\theta P_{l-1}^m(\cos\theta) - \frac{l+m-1}{l-m} P_{l-2}^m(\cos \right. \\ &= \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} \frac{2l-1}{l-m} e^{im\phi}\cos\theta P_{l-1}^m(\cos\theta) \end{aligned}$$

$$-\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} \frac{l+m-1}{l-m} e^{im\phi} P_{l-2}^m(\cos\theta) \tag{D3}$$

Replacing in this equation the expressions for $Y_{l-1}^m(\theta,\phi)$ and $Y_{l-2}^m(\theta,\phi)$ lead to equation 36, after basic algebra.

ii) $l > 0$ and $m = l - 1$

For those values of l and m, equation D2 remains valid, but with the second term dropped:

$$P_l^{l-1}(x) = (2l-1)xP_{l-1}^{l-1}(x) \tag{D4}$$

Replacing into D1, we get,

$$\begin{aligned} Y_l^{l-1}(\theta,\phi) &= \sqrt{\frac{2l+1}{4\pi(2l-1)!}}(2l-1)\cos\theta e^{i(l-1)\phi} P_{l-1}^{l-1}(\cos\theta) \\ &= \sqrt{2l+1}\cos\theta\sqrt{\frac{2l-1}{4\pi(2l-2)!}} e^{i(l-1)\phi} P_{l-1}^{l-1}(\cos\theta) \\ &= \sqrt{2l+1}\cos\theta Y_{l-1}^{l-1}(\theta,\phi) \end{aligned} \tag{D5}$$

which concludes the proof of equation 37.

iii) $l > 0$ and $m = l$

For those values of l and m, the Legendre polynomials satisfy

$$P_l^l(x) = -(2l-1)\sqrt{1-x^2}P_{l-1}^{l-1}(x) \tag{D6}$$

Replacing into D1, we get,

$$\begin{aligned} Y_l^l(\theta,\phi) &= -\sqrt{\frac{2l+1}{4\pi(2l)!}}(2l-1)\sin\theta e^{il\phi} P_{l-1}^{l-1}(\cos\theta) \\ &= -\sqrt{\frac{2l+1}{2l}} e^{i\phi}\sin\theta\sqrt{\frac{2l-1}{4\pi(2l-2)!}} e^{i(l-1)\phi} P_{l-1}^{l-1}(\cos\theta) \\ &= -\sqrt{\frac{2l+1}{2l}} e^{i\phi}\sin\theta Y_{l-1}^{l-1}(\theta,\phi) \end{aligned} \tag{D7}$$

which concludes the proof of equation 37.